# PATENT COOPERATION TREATY

From the INTERNATIONAL BUREAU

Intel

# PCT

NOTIFICATION CONCERNING TRANSMITTAL OF COPY OF INTERNATIONAL APPLICATION AS PUBLISHED OR REPUBLISHED

To:

VINCENT, Lester, J.
Blakely, Sokoloff, Tayor & Zafman
12400 Wilshire Boulevard
7th Floor
Los Angeles, CA 90025
ETATS-UNIS D'AMERIQUE



| | |
|---|---|
| Date of mailing *(day/month/year)*<br>03 November 2005 (03.11.2005) | |
| Applicant's or agent's file reference<br>P18443PCT | **IMPORTANT NOTICE** |

| International application No. | International filing date *(day/month/year)* | Priority date *(day/month/year)* |
|---|---|---|
| PCT/US2004/021015 | 29 June 2004 (29.06.2004) | 31 March 2004 (31.03.2004) |

Applicant: INTEL CORPORATION

The International Bureau transmits herewith the following documents:

☑ copy of the international application as published by the International Bureau on 03 November 2005 (03.11.2005) under No. WO 2005/104486

☐ copy of international application as republished by the International Bureau on under No. WO
For an explanation as to the reason for this republication of the international application, reference is made to INID codes (15), (48) or (88) (*as the case may be*) on the front page of the attached document.



| | |
|---|---|
| The International Bureau of WIPO<br>34, chemin des Colombettes<br>1211 Geneva 20, Switzerland | Authorized officer<br>Athina Nickitas-Etienne |

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 November 2005 (03.11.2005)

PCT

(10) International Publication Number
WO 2005/104486 A1

(51) International Patent Classification[7]: H04L 29/06, G06F 13/28

(21) International Application Number: PCT/US2004/021015

(22) International Filing Date: 29 June 2004 (29.06.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/815,895 31 March 2004 (31.03.2004) US

(71) Applicant (*for all designated States except US*): **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors: **VASUDEVAN, Anil,**; 12849 NW Marshall Court, Portland, OR 97229 (US). **BELL, Dennis**; 8160 SW 152nd Avenue, Beaverton, OR 97007 (US). **MINTURN, David**; 17125 Chehalem Way, Hillsboro, OR 97123 (US). **SEN, Sujoy**; 15947 NW Wismer Drive, Portland, OR 97229 (US).

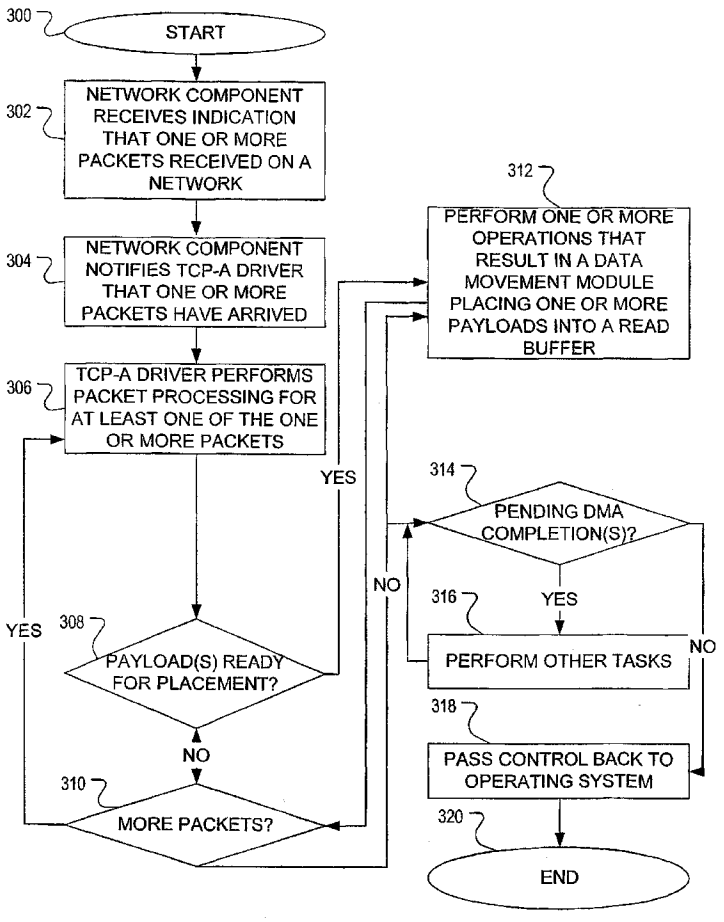
(74) Agents: **VINCENT, Lester, J.** et al.; Blakely, Sokoloff, Tayor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH,

*[Continued on next page]*

(54) Title: ACCELERATED TCP (TRANSPORT CONTROL PROTOCOL) STACK PROCESSING




(57) Abstract: In one embodiment, a method is provided. The method of this embodiment provides receiving an indication on a network component that one or more packets have been received from a network; the network component notifying a TCP-A (transport control protocol - accelerated) driver that the one or more packets have arrived; a TCP-A driver performing packet processing for at least one of the one or more packets; and the TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

WO 2005/104486 A1

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— *with international search report*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

# ACCELERATED TCP (TRANSPORT CONTROL PROTOCOL) STACK PROCESSING

## FIELD

**[0001]** Embodiments of this invention relate to accelerated TCP (Transport Control Protocol) stack processing.

## BACKGROUND

**[0002]** Networking has become an integral part of computer systems. Advances in network bandwidths, however, have not been fully utilized due to overhead that may be associated with processing protocol stacks. Overhead may result from bottlenecks in the computer system from using the core processing module of a host processor to perform slow memory access functions such as data movement, as well as host processor stalls related to data accesses missing the host processor caches. A protocol stack refers to a set of procedures and programs that may be executed to handle packets sent over a network, where the packets may conform to a specified protocol. For example, TCP/IP (Transport Control Protocol/Internet Protocol) packets may be processed using a TCP/IP stack.

**[0003]** One approach to reducing overhead is to partition protocol stack processing. For example, TCP/IP stack processing may be offloaded onto a TCP/IP offload engine (hereinafter "TOE"). In TOE, the entire TCP/IP stack may be offloaded onto a networking component, such as a MAC (media access control) component, of an I/O subsystem, such as a NIC (network interface card). However, a TOE may not be scalable to support a large number of connections due to the memory requirements associated with storing contexts associated with

these connections.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0004]** Embodiments of the present invention are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

**[0005]** FIG. 1 illustrates a network.

**[0006]** FIG. 2 illustrates a system according to one embodiment.

**[0007]** FIG. 3 is a flowchart illustrating a method according to one embodiment.

**[0008]** FIG. 4 is a flowchart illustrating a method according to another embodiment.

## DETAILED DESCRIPTION

**[0009]** Examples described below are for illustrative purposes only, and are in no way intended to limit embodiments of the invention. Thus, where examples may be described in detail, or where a list of examples may be provided, it should be understood that the examples are not to be construed as exhaustive, and do not limit embodiments of the invention to the examples described and/or illustrated.

**[0010]** FIG. 1 illustrates a network 100 in which embodiments of the invention may operate. Network 100 may comprise a plurality of nodes 102A, ...

102N, where each of nodes 102A, ... 102N may be communicatively coupled together via a communication medium 104. As used herein, components that are "communicatively coupled" means that the components may be capable of communicating with each other via wirelined (e.g., copper wires), or wireless (e.g., radio frequency) means. Nodes 102A . . . 102N may transmit and receive sets of one or more signals via medium 104 that may encode one or more packets.

**[0011]** As used herein, a "packet" means a sequence of one or more symbols and/or values that may be encoded by one or more signals transmitted from at least one sender to at least one receiver. As used herein, a "communication medium" means a physical entity through which electromagnetic radiation may be transmitted and/or received. Communication medium 104 may comprise, for example, one or more optical and/or electrical cables, although many alternatives are possible. For example, communication medium 104 may comprise air and/or vacuum, through which nodes 102A . . . 102N may wirelessly transmit and/or receive sets of one or more signals.

**[0012]** In network 100, one or more of the nodes 102A . . . 102N may comprise one or more intermediate stations, such as, for example, one or more hubs, switches, and/or routers; additionally or alternatively, one or more of the nodes 102A . . . 102N may comprise one or more end stations. Also additionally or alternatively, network 100 may comprise one or more not shown intermediate stations, and medium 104 may communicatively couple together at least some of the nodes 102A . . . 102N and one or more of these intermediate stations. Of course, many alternatives are possible.

**[0013]** At least one of nodes 102A, ..., 102N may comprise system 200, as

illustrated in FIG. 2. System 200 may comprise host processor 202, host memory 204, bus 206, and chipset 208. (System 200 may comprise more than one host processor 202, host memory 204, bus 206, and chipset 208, or other types of processors, memories, and busses; however, the former are illustrated for simplicity of discussion, and are not intended to limit embodiments of the invention.) Host processor 202, host memory 204, bus 206, and chipset 208 may be comprised in a single circuit board, such as, for example, a system motherboard 218.

**[0014]** Host processor 202 may comprise a core processing module and other support modules that interface with other system elements. For example, a support module may include a bus unit that communicates with a memory controller on system 200. Host processor 202 may comprise, for example, an Intel® Pentium® microprocessor that is commercially available from the Assignee of the subject application. Of course, alternatively, host processor 202 may comprise another type of microprocessor, such as, for example, a microprocessor that is manufactured and/or commercially available from a source other than the Assignee of the subject application, without departing from embodiments of the invention.

**[0015]** Host processor 202 may be communicatively coupled to chipset 208. Chipset 208 may comprise a host bridge/hub system that may couple host processor 202 and host memory 204 to each other and to bus 206. Chipset 208 may also include an I/O bridge/hub system (not shown) that may couple the host bridge/bus system to bus 206. Chipset 208 may comprise one or more integrated circuit chips, such as those selected from integrated circuit chipsets commercially

available from the Assignee of the subject application (e.g., graphics memory and I/O controller hub chipsets), although other one or more integrated circuit chips may also, or alternatively, be used.

**[0016]** Bus 206 may comprise a bus that complies with the Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.2, December 18, 1998 available from the PCI Special Interest Group, Portland, Oregon, U.S.A. (hereinafter referred to as a "PCI bus"). Alternatively, bus 106 instead may comprise a bus that complies with the PCI-X Specification Rev. 1.0a, July 24, 2000, (hereinafter referred to as a "PCI-X bus"), or a bus that complies with the PCI-E Specification Rev. PCI-E (hereinafter referred to as a "PCI-E bus"), as specified in "The PCI Express Base Specification of the PCI Special Interest Group", Revision 1.0a, both available from the aforesaid PCI Special Interest Group, Portland, Oregon, U.S.A. Also, alternatively, bus 106 may comprise other types and configurations of bus systems.

**[0017]** System 200 may additionally comprise circuitry 216. Circuitry 216 may comprise one or more circuits to perform one or more operations described herein as being performed by TCP-A (Transport Control Protocol – Accelerated) driver 222 and/or network component 212. Circuitry 216 may be hardwired to perform the one or more operations, and/or may execute machine-executable instructions to perform these operations. For example, circuitry 216 may comprise memory 236 that may store machine-executable instructions 226 that may be executed by circuitry 216 to perform these operations. Instead of being comprised in host processor 202, or chipset 208, some or all of circuitry 216 may be comprised in a circuit card (not shown), and/or other structures, systems, and/or

devices that may be, for example, comprised in motherboard 218, and/or communicatively coupled to bus 206, and may exchange data and/or commands with one or more other components in system 200. Circuitry 216 may comprise, for example, one or more digital circuits, one or more analog circuits, one or more state machines, programmable circuitry, and/or one or more ASIC's (Application-Specific Integrated Circuits).

**[0018]** System 200 may additionally comprise one or more memories to store machine-executable instructions 226 capable of being executed, and/or data capable of being accessed, operated upon, and/or manipulated by circuitry, such as circuitry 216. For example, these one or more memories may include host memory 204, or memory 236. One or more memories 204, 236 may, for example, comprise read only, mass storage, random access computer-readable memory, and/or one or more other types of machine-readable memory. The execution of program instructions 226 and/or the accessing, operation upon, and/or manipulation of data by circuitry 216 may result in, for example, circuitry 216 carrying out some or all of the operations described herein as being carried out by various hardware and/or software components in system 200.

**[0019]** For example, machine-executable instructions 226 may comprise a set of instructions for an application 218; a set of instructions for operating system 220; a set of instructions for TCP-A driver 222; and/or a set of instructions for DMA driver 224. In one embodiment, circuitry 216 of host processor 202 may execute machine-executable instructions 226 for TCP-A driver 222, for DMA driver 224, and for operating system 220. Machine-executable instructions 226 may execute in memory by circuitry 216, such as in host processor 202, and/or by

circuitry 216 in general.

**[0020]** A method according to one embodiment is illustrated in the flowchart of FIG. 3 with reference to system 200 of FIG. 2. The method begins at block 300, and continues to block 302 where network component 212 may receive an indication that one or more packets 228 (only one shown) have been received from network 100. Each packet 228 may comprise a header 230 and a payload 232. For each packet 228, network component 212 may split header 230 and payload 232 from packet 228, and post each 230, 232 to one or more post buffers 214A, 214B. In one embodiment, header 230 may be posted to a first buffer such as post buffer 214A, and payload 232 may be posted to a second buffer such as post buffer 214B. The one or more packets 228 may be comprised in one or more groups, and each group of packets 228 may be transmitted and/or received over a connection. The one or more packets 228 may be received in response to a read data request from, for example, application 218.

**[0021]** "Application" refers to one or more programs that use the network. An application 218 may comprise, for example, a web browser, an email serving application, a file serving application, or a database application. In conjunction with a read data request, application 218 may designate destination read buffer 214C where application 218 may access the requested data. In conjunction with a transmit data request, application 218 may write data to be transmitted to source buffer 214D.

**[0022]** "Network component" refers to any combination of hardware and/or software on an I/O (input/output) subsystem that may process one or more packets sent and/or received over a network. In one embodiment, I/O subsystem

238 may comprise, for example, a NIC (network interface card), and network component 212 may comprise, for example, a MAC (media access control) layer of the Data Link Layer as defined in the Open System Interconnection (OSI) model for networking protocols. The OSI model is defined by the International Organization for Standardization (ISO) located at 1 rue de Varembé, Case postale 56 CH-1211 Geneva 20, Switzerland.

**[0023]** A "connection" as used herein refers to a logical pathway to facilitate communications between a first node on a network and a second node on the network. A connection may facilitate communications for one or more transactions between the first node and the second node. A "transaction" refers to a request to send or receive data, such as may be initiated by an application, such as application 218, on a system, such as system 200. Each connection may be associated with a protocol context. As used herein, "protocol context" refers to information about a connection. For example, the information may comprise the sequence number of the last packet sent/received, and amount of memory available.

**[0024]** At block 304, network component 212 may notify TCP-A driver that one or more packets 228 have arrived. In one embodiment, network component 212 may notify TCP-A driver 222 by notifying operating system 220 in accordance with an interrupt moderation scheme. An interrupt moderation scheme refers to a condition where an interrupt may be asserted for every n packets received by network component 212. Thus, if network component 212 receives n or more packets, network component 212 may notify operating system 220 that packets have arrived. Likewise, if network component 212 receives less than n packets,

network component 212 may instead wait until more packets are received before notifying operating system 220. In one embodiment, operating system 220 may then notify TCP-A driver 222 that packets are ready to be processed.

**[0025]** At block 306, TCP-A driver 222 may perform packet processing for at least one of the one or more packets. Packet processing may be performed by the TCP-A driver 222 retrieving header 230 from post buffer 214A, parsing the header 230 to determine the protocol context associated with the current connection, and performing TCP protocol compliance. TCP protocol compliance may comprise, for example, verifying the sequence number of a received packet to ensure that the packet is within a range of numbers that was agreed upon between the communicating nodes; verifying the payload size to ensure that the packet is within a range of sizes that was agreed upon between the communicating nodes; ensuring that the header structure conforms to the protocol; and ensuring that the timestamps are within an expected time range.

**[0026]** TCP-A driver 222 may fetch a next header to process prior to completing the processing of a current header. This may ensure that the next header is available in the host processor's caches (not shown) before the TCP-A driver 222 is ready to perform TCP processing on it, thereby reducing host processor stalls. The method may continue to block 308.

**[0027]** In one embodiment, TCP-A driver 222 may additionally determine if a connection associated with a packet is to be accelerated prior to performing packet processing. TCP-A driver 222 may accelerate select connections. Select connections may comprise, for example, connections that are long-lived, or which comprise large data. If TCP-A driver 222 determines that network connection is to

be accelerated, TCP-A driver 222 may perform packet processing as described at block 306. If TCP-A driver 222 determines that network connection is not to be accelerated, the method may continue to block 318.

**[0028]** At block 308, TCP-A driver 222 may determine if one or more payloads 232 placed in post buffer 214B are ready for placement. A payload 232 may be ready for placement if, for example, the corresponding header has been successfully processed, and a read buffer, such as read buffer 214C, has been designated. If at block 308, payload 232 is not ready for placement, the method may continue to block 310. In one embodiment, TCP-A driver 222 may determine if there are one or more payloads 232 ready for placement at anytime. For example, if it is determined that payload 232 is not ready for placement, TCP-A driver 222 may wait for some period of time before it makes this determination again. Where payload 232 cannot be placed because a read buffer 214C does not exist, for example, TCP-A driver 222 may alternatively or additionally at anytime indicate to operating system 220 the presence of payload 232 ready to be placed. Operating system 220 may then designate a buffer, or may ask application 218 to designate a buffer. If there are one or more payloads ready for placement, the method may continue to block 312.

**[0029]** At block 310, TCP-A driver 222 may determine if there are more packets 228 to process, for example in post buffer 214A, of the n packets for the current interrupt. If there are more packets 228 to process, the method reverts to block 306. If there are no more packets 228, and one or more packets 228 have not been previously placed, and are ready for placement, the method may continue to block 312. If there are no more packets 228 to process, and there are

no previous packets 228 to place, the method may continue to block 314.

**[0030]** At block 312, TCP-A driver 222 may perform one or more operations that result in a data movement module placing one or more corresponding payloads 232 into a read buffer, such as read buffer 214C. As used herein, a "data movement module" refers to a module for moving data from a source to a destination without using the core processing module of a host processor, such as host processor 202. A data movement module may comprise, for example, a DMA engine as described below.

**[0031]** In one embodiment, for example, TCP-A driver 222 may send a request to DMA driver 224, and DMA driver 224 may schedule a request with DMA engine 210 to write the one or more payloads 232 from post buffer 214B to read buffer 214C. In another embodiment, TCP-A driver 222 may directly program DMA engine 210 to write the one or more payloads 232 from post buffer 214B to read buffer 214C. DMA driver 224 may be a standalone driver, or part of some other driver, such as TCP-A driver 222. Rather than being part of chipset 208, DMA engine 210 may be a support module of host processor 202. By using the DMA engine for placement of data, host processor 202 may be freed from the overhead of performing data movements, which may result in the host processor 202 running at much slower memory speeds compared to the core processing module speeds. Following the DMA engine 210 scheduling, the method may revert to block 310 to determine if there are additional packets 228 to process.

**[0032]** At block 314, TCP-A driver 222 may determine if there are any pending DMA completions for the current interrupt. Alternatively, TCP-A driver 222 may look for DMA completions at anytime. A "pending completion" as used

herein refers to the completion of a request. In one embodiment, a pending completion refers to the completion of a request to DMA engine 210 to write one or more payloads 232. If, at block 314, there are one or more pending DMA completions for the current interrupt, the method may continue to block 316. If at block 314 there are no pending DMA completions for the current interrupt, the method may continue to block 318.

**[0033]** At block 316, TCP-A driver 222 may perform other tasks. Other tasks may include looking for more packets in a subsequent interrupt, setting up the DMA engine 210 to issue an interrupt upon completion of a last queued task for the current interrupt, or other housekeeping, such as transmitting data, and performing TCP timer related tasks.

**[0034]** At block 318, TCP-A driver 222 may pass control back to operating system 220. If one or more packets 228 have still not been processed, operating system 220 may notify a TCP driver (not shown) rather than TCP-A driver 222, where the TCP driver may perform TCP stack processing by performing packet processing, and by using the core processing module of host processor 202 to perform data transfers. If all packets 228 have been processed, operating system 220 may wait for a next interrupt.

**[0035]** The method may end at block 320.

**[0036]** A method according to another embodiment is illustrated in FIG. 4. The method begins at block 400 and continues to block 402 where operating system 220 may receive a request from application 218 to transmit data 234 placed in buffer 214D. Operating system 220 may perform preliminary checks on

data 234. Preliminary checks may include, for example, obtaining the associated protocol context. In a TCP/IP connection, for example, protocol context may comprise packet sequence numbers to identify the order of the packets, buffer addresses of buffers used to store data, and timer/timestamp information for retransmissions.

**[0037]** At block 404, operating system 220 may notify TCP-A driver 222 that there is data 234 to be transmitted from buffer 214D.

**[0038]** At block 406, TCP-A driver 222 may perform one or more operations that result in data 234 being transmitted to network component 212. For example, these one or more operations may include TCP-A driver 222 programming DMA engine 210 to transmit data 234 from source buffer 214D to network component 212. Alternatively, TCP-A driver 222 may queue a buffer, such as queued buffer 214E, to network component 212, where network component 212 may instead read data 234 from queued buffer 214E. Source buffer 214D may be designated by application 218, for example, and queued buffer 214E may be designated by network component 212, for example.

**[0039]** In one embodiment, TCP-A driver 222 may program DMA engine 210 to transmit data if the data is small, and TCP-A driver 222 may queue a buffer, such as queued buffer 214E, if the data is large. As used herein, "queuing a buffer" means to notify a component that there is a buffer from which it can access data. For example, TCP acknowledgment packets to acknowledge receipt of packets may typically be relatively small-sized packets, and may be sent by TCP-A driver 222 to network component 212 by TCP-A driver 222 programming DMA engine 210 to transmit data 234. As another example, storage applications

that send large files over the network may be relatively large, and may therefore be sent by TCP-A driver 222 to network component 212 by queuing buffer 214E.

**[0040]** At block 408, in response to receiving the data, network component 212 may create one or more packets for transmission by packetizing the data. In one embodiment, network component 212 may packetize data by performing segmentation on the data. "Segmentation" refers to breaking the data into smaller pieces for transmission. In one embodiment, network component 212 may comprise a MAC, and segmentation may be referred to as a large send offload, wherein MAC frames may be created for transmission of data 234 over the network. Network component 212 may receive data directly from TCP-A driver 222, or by accessing queued buffer 214E.

**[0041]** The method may end at block 410. Thereafter, operating system 220 may send a completion notification to application 218. Furthermore, source buffer 214D may be returned to application 218, and application may use the buffer for other purposes.

**[0042]** Embodiments of the present invention may be provided, for example, as a computer program product which may include one or more machine-readable media having stored thereon machine-executable instructions that, when executed by one or more machines such as a computer, network of computers, or other electronic devices, may result in the one or more machines carrying out operations in accordance with embodiments of the present invention. A machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs (Compact Disc-Read Only Memories), and magneto-optical disks, ROMs (Read Only Memories), RAMs (Random Access Memories),

EPROMs (Erasable Programmable Read Only Memories), EEPROMs (Electrically Erasable Programmable Read Only Memories), magnetic or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing machine-executable instructions.

**[0043]** Moreover, embodiments of the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of one or more data signals embodied in and/or modulated by a carrier wave or other propagation medium via a communication link (e.g., a modem and/or network connection). Accordingly, as used herein, a machine-readable medium may, but is not required to, comprise such a carrier wave.

Conclusion

**[0044]** Therefore, in one embodiment, a method may comprise receiving an indication on a network component that one or more packets have been received from a network; the network component notifying a TCP-A (transport control protocol – accelerated) driver that the one or more packets have arrived; a TCP-A driver performing packet processing for at least one of the one or more packets; and the TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

**[0045]** Embodiments of the invention may significantly reduce TCP/IP processing overhead that may result from using the core processing module of a host processor. TCP/IP processing may be accelerated by using a data

movement module, such as a DMA engine, to move data from one buffer to another buffer. Since the core processing module of a host processor may be bypassed using a DMA engine, slow memory access speeds may be avoided. Furthermore, TCP/IP processing performed on the host processor may scale better than TOE processing because the number of contexts is not limited by TOE memory.

**[0046]** In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made to these embodiments without departing therefrom. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

**WHAT IS CLAIMED IS:**

1. A method comprising:

receiving an indication on a network component that one or more packets have been received from a network;

the network component notifying a TCP-A (transport control protocol – accelerated) driver that the one or more packets have arrived;

a TCP-A driver performing packet processing for at least one of the one or more packets; and

the TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

2. The method of claim 1, additionally comprising, in response to receiving an indication on a network component that one or more packets have been received from the network, the network component:

splitting each of the one or more packets into a header and a payload; and

posting each of the header and payload to one or more post buffers.

3. The method of claim 2, wherein the TCP-A driver performs packet processing by processing each of the headers, the method additionally comprising fetching a next header of the one or more headers prior to completing the processing of the current header.

4. The method of claim 1, wherein said performing one or more operations

that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer comprises sending a request to a data movement module driver to write the one or more corresponding payloads to the read buffer.

5. The method of claim 1, wherein said TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer comprises programming the data movement module to write the one or more corresponding payloads to the read buffer.

6. The method of claim 1, wherein the data movement module comprises a DMA (direct memory access) engine.

7. The method of claim 6, wherein the DMA engine resides on a chipset.

8. The method of claim 6, wherein the DMA engine resides on a host processor as a support module.

9. The method of claim 1, additionally comprising:

receiving a request on an operating system to transmit data over the network;

the operating system notifying the TCP-A driver that there is data to be transmitted;

the TCP-A driver performing one or more operations that result in the data being transmitted to the network component;

in response to receiving the data, the network component creating one or more packets for transmission by packetizing the data; and

the network component transmitting the one or more packets over the network.

10. An apparatus comprising:

a network component capable of:

receiving an indication on a network component that one or more packets have been received from a network; and

notifying a TCP-A (transport control protocol – accelerated) driver that the one or more packets have arrived; and

a TCP-A driver capable of:

performing packet processing for at least one of the one or more packets; and

the TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

11. The apparatus of claim 10, additionally comprising an operating system capable of:

receiving a request to transmit data over the network; and

notifying the TCP-A driver that data is ready to be transmitted;

wherein:

the TCP-A driver is capable of performing one or more operations that result in the data being transmitted to the network component; and

the network component is capable of:

creating one or more packets for transmission by packetizing the data in response to receiving the data; and

transmitting the one or more packets over the network.

12. The apparatus of claim 10, wherein in response to receiving an indication on a network component that one or more packets have been received from the network, the network component is additionally capable of:

splitting each of the one or more packets into a header and a payload; and

posting each of the header and payload to one or more post buffers.

13. The apparatus of claim 12, wherein the TCP-A driver performs packet processing by processing each of the headers, and the TCP-A driver is additionally capable of fetching a next header of the one or more headers prior to completing the processing of the current header.

14. A system comprising:

a chipset having a DMA (direct memory access) engine, the chipset communicatively coupled to a TCP-A (Transport Control Protocol – Accelerated) driver of a processor and to a network component;

the network component capable of:

receiving an indication that one or more packets have been received from a network; and

notifying the TCP-A (transport control protocol – accelerated) driver that the one or more packets have arrived; and

the TCP-A driver of the processor capable of:

performing packet processing for at least one of the one or more packets; and

performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

15. The system of claim 14, additionally comprising an operating system of the processor capable of:

receiving a request to transmit data over the network; and

notifying the TCP-A driver that data is ready to be transmitted;

wherein:

the TCP-A driver is capable of performing one or more operations that result in the data being transmitted to a network component; and

the network component is capable of:

creating one or more packets for transmission by packetizing the data in response to receiving the data; and

transmitting the one or more packets over the network.

16. The system of claim 14, wherein in response to receiving an indication on a network component that one or more packets have been received from the network, the network component is additionally capable of:

splitting each of the one or more packets into a header and a payload; and

posting each of the header and payload to one or more post buffers.

17. The system of claim 16, wherein the TCP-A driver performs packet processing by processing each of the headers, and the TCP-A driver is additionally capable of fetching a next header of the one or more headers prior to completing the processing of the current header.

18. A machine-readable medium having stored thereon instructions, the instructions when executed by a machine, result in the following:

receiving an indication on a network component that one or more packets have been received from a network;

the network component notifying a TCP-A (transport control protocol – accelerated) driver that the one or more packets have arrived;

a TCP-A driver performing packet processing for at least one of the one or more packets; and

the TCP-A driver performing one or more operations that result in a data

movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer.

19. The machine-readable medium of claim 18, wherein in response to receiving an indication on a network component that one or more packets have been received from the network, the instructions additionally result in:

splitting each of the one or more packets into a header and a payload; and

posting each of the header and payload to one or more post buffers.

20. The machine-readable medium of claim 19, wherein the TCP-A driver performs packet processing by processing each of the headers, the instructions additionally result in fetching a next header of the one or more headers prior to completing the processing of the current header.

21. The machine-readable medium of claim 18, wherein the instructions that result in performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer additionally result in sending a request to a data movement module driver to write the one or more corresponding payloads to the read buffer.

22. The machine-readable medium of claim 18, wherein the instructions that result in said TCP-A driver performing one or more operations that result in a data movement module placing one or more corresponding payloads of the at least one of the one or more packets into a read buffer additionally result in programming the data movement module to write the one or more

corresponding payloads to the read buffer.

23. The machine-readable medium of claim 18, wherein the data movement module comprises a DMA (direct memory access) engine.

24. The machine-readable medium of claim 24, wherein the DMA engine resides on a chipset.

25. The machine-readable medium of claim 24, wherein the DMA engine resides on a host processor as a support module.

26. The machine-readable medium of claim 18, the instructions additionally result in:

receiving a request on an operating system to transmit data over the network;

the operating system notifying the TCP-A driver that there is data to be transmitted;

the TCP-A driver performing one or more operations that result in the data being transmitted to the network component;

in response to receiving the data, the network component creating one or more packets for transmission by packetizing the data; and

the network component transmitting the one or more packets over the network.

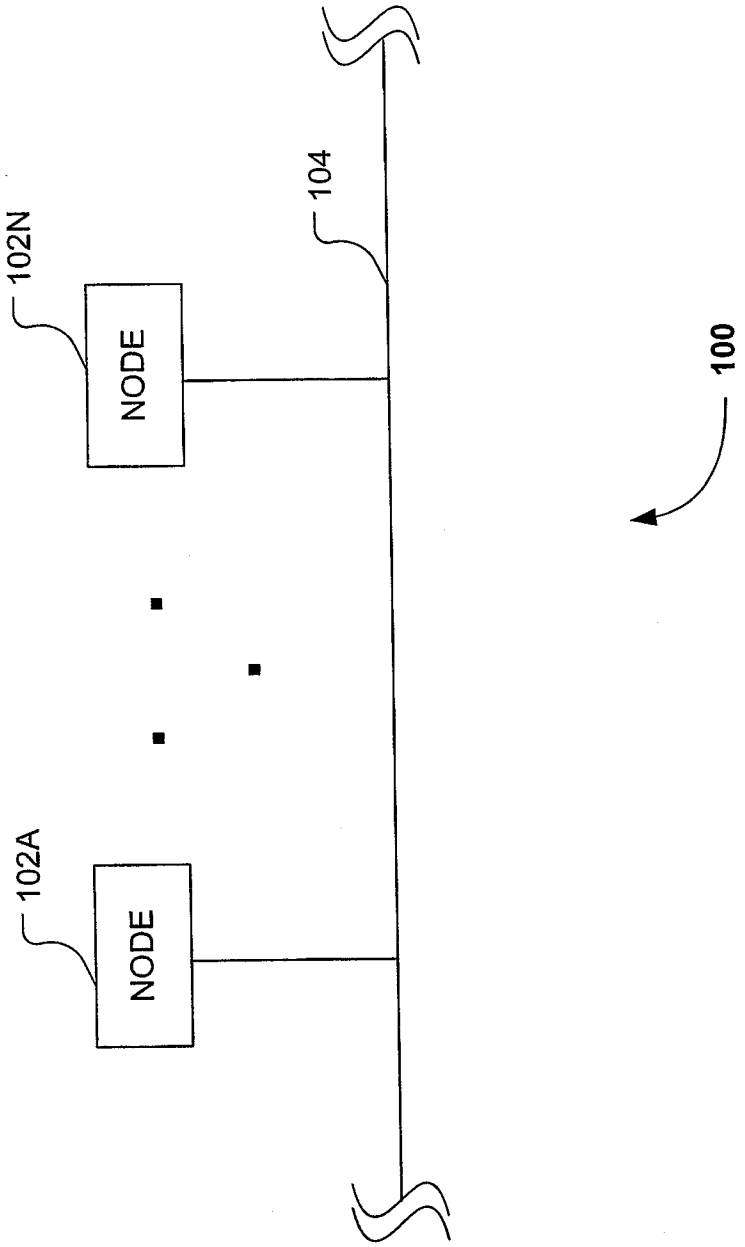



FIG. 1

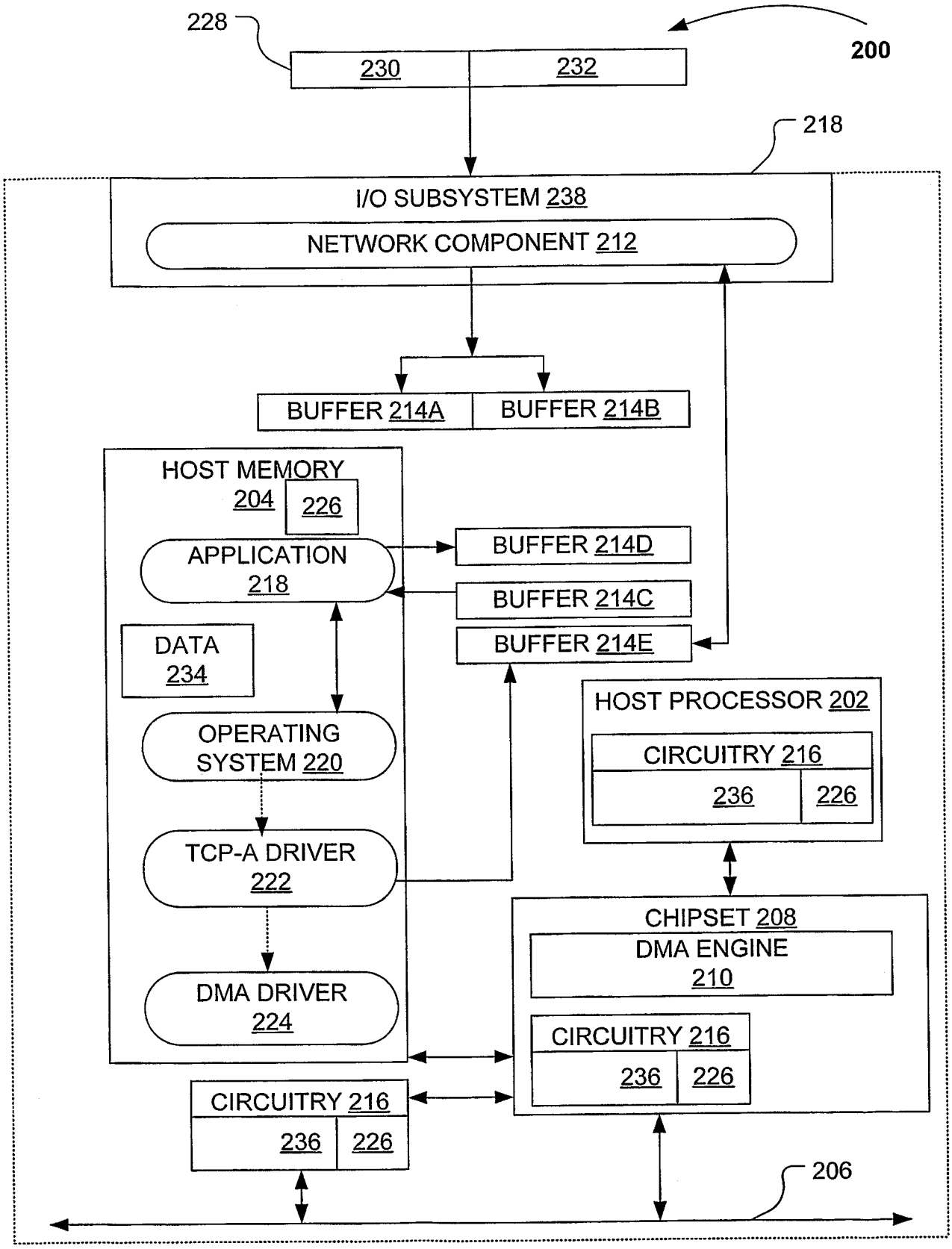

228
230
232
200
218
I/O SUBSYSTEM 238
NETWORK COMPONENT 212
BUFFER 214A
BUFFER 214B
HOST MEMORY 204
226
APPLICATION 218
BUFFER 214D
BUFFER 214C
BUFFER 214E
DATA 234
HOST PROCESSOR 202
CIRCUITRY 216
236
226
OPERATING SYSTEM 220
TCP-A DRIVER 222
CHIPSET 208
DMA ENGINE 210
DMA DRIVER 224
CIRCUITRY 216
236
226
CIRCUITRY 216
236
226
206

FIG. 2

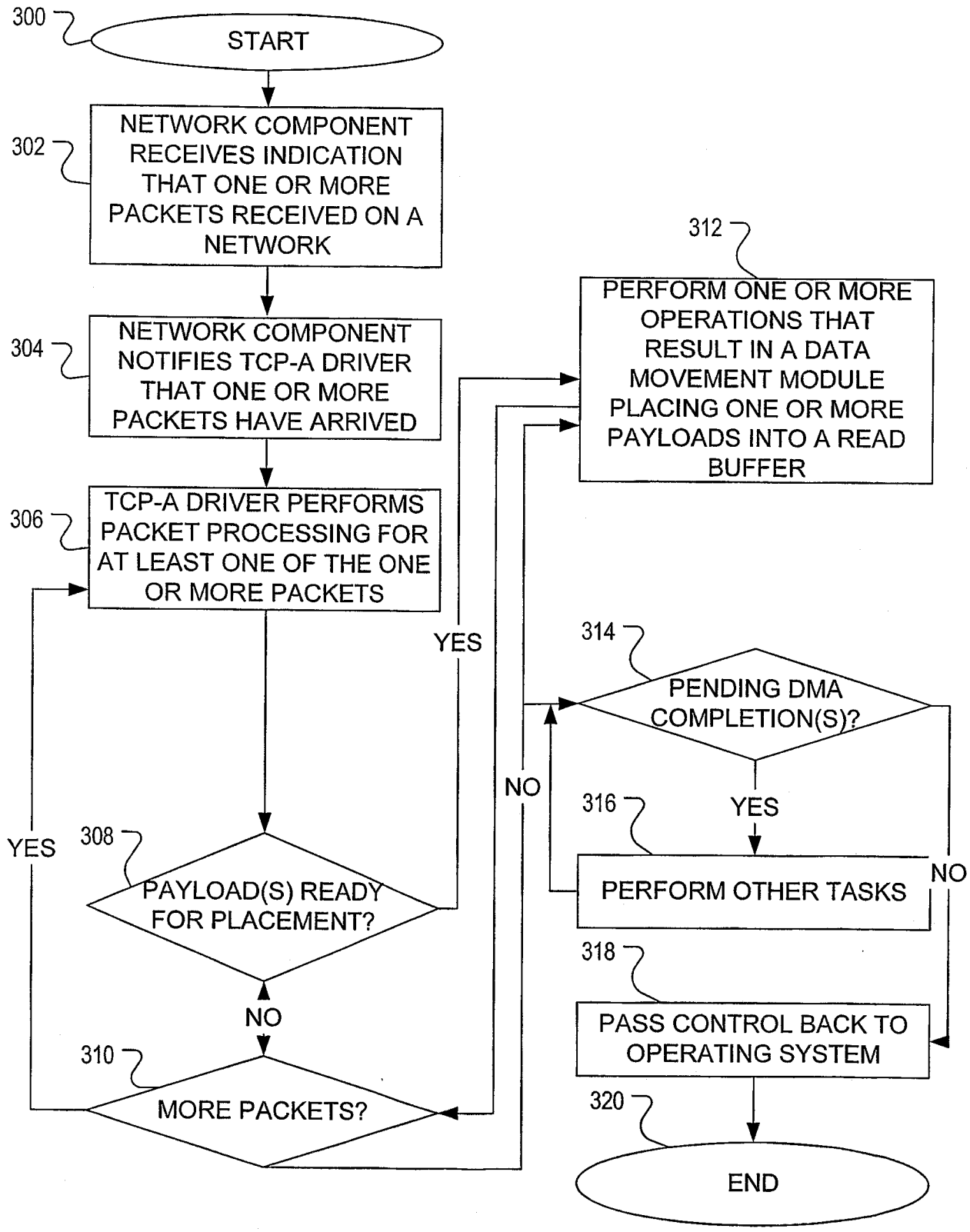

300
START
302
NETWORK COMPONENT RECEIVES INDICATION THAT ONE OR MORE PACKETS RECEIVED ON A NETWORK
304
NETWORK COMPONENT NOTIFIES TCP-A DRIVER THAT ONE OR MORE PACKETS HAVE ARRIVED
306
TCP-A DRIVER PERFORMS PACKET PROCESSING FOR AT LEAST ONE OF THE ONE OR MORE PACKETS
308
PAYLOAD(S) READY FOR PLACEMENT?
YES
NO
310
MORE PACKETS?
YES
NO
312
PERFORM ONE OR MORE OPERATIONS THAT RESULT IN A DATA MOVEMENT MODULE PLACING ONE OR MORE PAYLOADS INTO A READ BUFFER
314
PENDING DMA COMPLETION(S)?
YES
NO
316
PERFORM OTHER TASKS
318
PASS CONTROL BACK TO OPERATING SYSTEM
320
END

FIG. 3

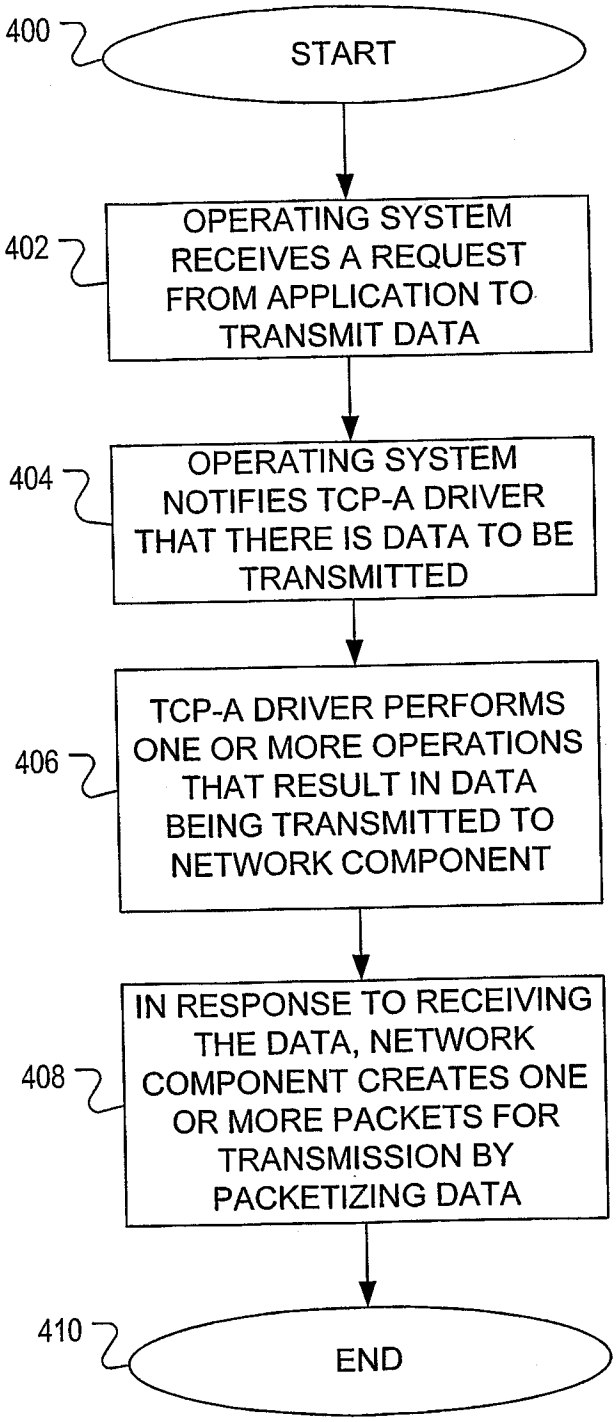

400
START
402
OPERATING SYSTEM RECEIVES A REQUEST FROM APPLICATION TO TRANSMIT DATA
404
OPERATING SYSTEM NOTIFIES TCP-A DRIVER THAT THERE IS DATA TO BE TRANSMITTED
406
TCP-A DRIVER PERFORMS ONE OR MORE OPERATIONS THAT RESULT IN DATA BEING TRANSMITTED TO NETWORK COMPONENT
408
IN RESPONSE TO RECEIVING THE DATA, NETWORK COMPONENT CREATES ONE OR MORE PACKETS FOR TRANSMISSION BY PACKETIZING DATA
410
END

FIG. 4